Г. Б. МАРАНДЖЯН

ОБ ОДНОМ МЕТОДЕ СИНТЕЗА ПРОГРАММ ЧИСЛОВЫХ ФУНКЦИЙ

В данной статье описаны основные черты разработанной в Вычислительном центре АН Армянской ССР и Ереванского государственного университета экспериментальной системы, предназначенной для автоматизированного синтеза программ, решающих задачи одного специального вида. Мы не будем пытаться сформулировать общую задачу автоматизированного синтеза программ — разные авторы по-разному подходят к этой задаче в зависимости от той конкретной цели, которая перед ними стоит и от того математического аппарата, которому отдают предпочтение. Различные подходы к синтезу программ можно найти в работах [1, 2, 4, 5, 8—11]. Упомянутый список работ, разумеется, не может претендовать на полноту, но и они демонстрируют большое разнообразие взглядов и подходов к решению задач этого типа.

Задача синтеза программ, рассматриваемая нами, ставит перед собой конкретную цель — вычисление функций целочисленного аргумента с целочисленными же значениями для случая, когда описание функции имеет специальный, хотя и достаточно общий вид. Один из подходов к решению задач такого вида содержится в работе [9]. Отличие нашего подхода, в частности, состоит в том, что мы не требуем, чтобы описание носило экстенсиональный характер. Имеется и ряд других отличий. Для решения задачи мы обратимся к способу построения функций, весьма типичному для работ по теории алгоритмов, а именно к построению с помощью второй теоремы С. К. Клини о рекурсии [3]. Для нашей цели удобен следующий вариант этой теоремы, который можно найти, например, в [7].

Теорема. *Для всякой геделевской нумерации* $\{\varphi_i\}$ *частично рекурсивных функций существует такая общерекурсивная функция* h, *что для любой общерекурсивной функции* f *имеет место соотношение*

$$\varphi_{fh(a)}(x_1, x_2, \ldots, x_m) \simeq \varphi_{h(a)}(x_1, x_2, \ldots, x_m),$$

где a *— какой-нибудь номер функции* fh *в нумерации* $\{\varphi_i\}$.

Рассмотрим простой пример, иллюстрирующий применение этой теоремы для построения алгоритма вычисления функции, заданной «неявным» образом. Пусть, например, требуется построить вычислимую функцию натурального аргумента, удовлетворяющую следующим условиям:

$$\alpha(x, y) = \begin{cases} 0, & \text{если } \beta(x, y) = 1; \\ 3 \cdot \alpha(x+1, y), & \text{если } \beta(x, y) = 2; \\ 2^{(M_3(S(\alpha, I_1^3, R(+, \delta))))(x, y)+1}, & \text{если } \beta(x, y) = 3, \end{cases} \quad (1)$$

где

$$\beta(x, y) = \begin{cases} 1, \text{ если } x \cdot y = 0, \text{ или } (x+y \text{ простое число} \\ \qquad \text{и взаимно просто с } y); \\ 2, \text{ если } x \text{ и } y \text{ не взаимно просты и } x \cdot y \neq 0; \\ 3, \text{ если } x \text{ и } y \text{ взаимно просты, но } x+y \\ \qquad \text{не простое число и } x \cdot y \neq 0. \end{cases}$$

$\delta(x, y, z, t) = x + t$; R обозначает операцию примитивной рекурсии, S — операцию суперпозиции, а M_3 — операцию минимизации, примененную по третьей переменной.

Задача синтеза программ для вычисления функции осмысленна, если, по меньшей мере, существует (быть может, частичная) функция, удовлетворяющая поставленным условиям. (Отметим, что условия, наложенные на функцию α, таковы, что α оказывается всюду определенной — в этом можно убедиться, применив теорему Дирихле о простых числах в арифметической прогрессии)

Поскольку решение ищется в классе функций, входящих в некоторую фиксированную нумерацию, то задача может быть переформулирована следующим образом: найти такое натуральное число i_0, чтобы было выполнено условие

$$\varphi_{i_0}(x, y) = \begin{cases} 0, & \text{если } \beta(x, y) = 1; \\ 3 \cdot \varphi_{i_0}(x+1, y), & \text{если } \beta(x, y) = 2; \\ 2^{(M_3(S(\varphi_{i_0}, I_1^3, R(+, \delta))))(x, y)+1}, & \text{если } \beta(x, y) = 3. \end{cases}$$

Определение искомой функции остается все еще неявным, но уже принадлежит конкретной формальной системе. Поскольку для любого фиксированного натурального числа i φ_i есть вполне определенная функция, то можно определить функцию α^* следующим образом:

$$\alpha^*(i, x, y) = \begin{cases} 0, & \text{если } \beta(x, y) = 1: \\ 2 \cdot \varphi_i(x+1, y), & \text{если } \beta(x, y) = 2; \\ 2^{M_3(S(\varphi_i, I_1^3, R(+, \delta))))(x, y)+1}, & \text{если } \beta(x, y) = 3. \end{cases}$$

Здесь функция α^* определена уже явным образом, поскольку в правой части она уже не встречается. Фиксируя различные значения параметра i, мы получим некоторую последовательность частично рекурсивных функций. Поскольку схема определения функции α^* остается при этом неизменной, то можно алгоритмически описать ту функцию, которая по любому значению параметра i вычисляет номер частично рекурсивной функции, получающейся при данном значении i, или, иначе, можно по-

строить такую общерекурсивную функцию f, что будет иметь место соотношение

$$\varphi_{f(i)}(x, y) \simeq \alpha^*(i, x, y).$$

Теперь, обозначив через a какой-нибудь номер функции fh в нумерации $\{\varphi_i\}$, а затем применив теорему С. К. Клини, упомянутую выше, получим

$$\varphi_{h(a)}(x,y)=\varphi_{fh(a)}(x, y)=\begin{cases}0, & \text{если } \beta(x, y)=1;\\ 3\cdot\varphi_{h(a)}(x+1, y), & \text{если } \beta(x, y)=2;\\ 2^{M_3(S(\varphi_{h(a)}, I_1^3, R(+, 3)))(x, y)+1}, & \text{если } \beta(x, y)=3\end{cases}$$

откуда следует, что $\varphi_{h(a)}$ является решением системы (1).

Исключительное удобство, поставляемое теоремой С. К. Клини, заключается в том, что мы почти автоматически получили нужный нам номер функции, а значит и алгоритм вычисления этой функции, в то время как без использования этой теоремы построение алгоритма было бы сложнее, так как потребовало бы вникания в подробности схемы задания функции и организации всего процесса вычисления. И если в приведенном случае алгоритм еще легко построить, то небольшое усложнение схемы приводит к значительно более сложной задаче организации алгоритма. Отметим еще одно немаловажное обстоятельство. Решение других аналогичных задач по построению алгоритма с помощью приведенной выше методики имеет практически тот же ход построения. Отличия в решении различных задач рассмотренного типа состоят лишь в том, что различными оказываются функции, играющие роль, аналогичную той, которую играла в приведенном примере функция f. Однако процесс вычисления функции f легко описать, если удобным образом организовать нумерацию, предусмотрев в ней легкость получения номеров функций, описываемых с помощью наиболее распространенных форм описания.

Попытка синтеза программ на принципе, буквально повторяющем конструкцию доказательства второй теоремы о рекурсии, сталкивается с определенными трудностями. Ясно, что для применения этой теоремы необходимо выбрать некоторую нумерацию, которую, очевидно, следует задавать с помощью базисных функций и некоторого набора порождающих операций. При этом выбор того или иного базиса, а также того или иного набора порождающих операций в значительной степени влияет на объем результирующих программ, а также приводит к функциям, играющим роль функции f в приведенном примере, которые имеют слишком большие геделевы номера, чтобы быть обработанными на реальной вычислительной машине. Для преодоления этой трудности мы не фиксируем раз и навсегда конкретную геделеву нумерацию. Строится некоторое «ядро» нумерации, которое надстраивается таким образом, чтобы, по возможности, облегчить решение поставленной конкретной задачи.

Принципиально неважно функции скольких переменных нумеруются, но в нашем случае их количество не превышает 9. Для базисных функций выделено 40 номеров от 1 до 40; номер 0 играет вспомогательную техническую роль. Номера разделены на два подмножества — $Б_0$ и $Б_1$. На номерах из списка $Б_0$ размещены функции, составляющие «ядро» базиса, то есть функции, необходимые, вообще говоря, почти в каждом случае из рассматриваемого класса задач. Заполнение $Б_1$ меняется от задачи к задаче. Характер заполнения $Б_1$ будет описан ниже. Таким образом, мы не связываем себя с какой бы то ни было конкретной нумерацией, а вместо этого каждый раз работаем с такой нумерацией, которая представляется наиболее естественной для того, чтобы решение задачи облегчалось в возможно большей степени.

В качестве основных порождающих операций выбраны: ветвление, имеющее вид

$$\beta(x_1, x_2, \ldots, x_m) = \begin{cases} \gamma_1(x_1, x_2, \ldots, x_m), & \text{если } \delta(x_1, x_2, \ldots, x_m) = 1; \\ \gamma_2(x_1, x_2, \ldots, x_m), & \text{если } \delta(x_1, x_2, \ldots, x_m) = 2; \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \gamma_p(x_1, x_2, \ldots, x_m), & \text{если } \delta(x_1, x_2, \ldots, x_m) = p, \end{cases}$$

и допустимое лишь как самая внешняя операция (хотя легко превратить эту операцию в допустимую на любом месте), примитивная рекурсия по любой из переменных и операция минимизации по любой из переменных. Предусмотрено легкое расширение списка допустимых операций пользователем, нуждающемся в каких-либо стандартных операциях, характерных для решения его задач.

После того, как становится ясно, функции какого числа переменных будут участвовать в построении, фиксируем это число и в дальнейшем будем обозначать его через k.

Обозначим через $c(t_1, t_2, \ldots, t_q)$ число, определяемое индуктивно следующим образом:

$$c(a, b) = \frac{(a+b)(a+b+1)}{2} + a;$$

$$c(t_1, t_2, \ldots, t_q) = c(t_1, c(t_2, \ldots, t_q)).$$

Номера небазисных функций определяются следующим образом:

1. Результат подстановки функций с номерами $a_1, a_2, \ldots, a_q$ на места $i_1, i_2, \ldots, i_q$ в функцию с номером b имеет номер

$$20 + 3 \cdot c(b, 0, \ldots, a_1, 0, \ldots, 0, a_2, 0, \ldots, a_q, 0, \ldots, 0).$$

Нули в приведенном выражении используются для указания того, что на этих местах не произведено изменений.

2. Результат применения операции примитивной рекурсии к функциям с номерами a и b по переменной с порядковым номером t получает номер

$$20 + 1 + 3 \cdot c(a, b, t),$$

3. Результат применения операции минимизации к переменной с номером t в функции с номером a получает номер

$$20 + 2 + 3 \cdot c(a, t).$$

Роль функции $h(t)$ при такой нумерации играет функция $c\ (4, t)$.

Теперь опишем процесс решения задачи синтеза.

Пусть дано неявное описание функции α в виде уравнения вида B

$$\alpha = P(\alpha, G_1, G_2, \ldots, G_p).$$

В правой части описания встречаются вхождения символа, обозначающего искомую функцию и ряда других, но уже заданных явным образом функций $G_1, G_2, \ldots, G_p$, а также схемы P, которая может содержать операции суперпозиции, примитивной рекурсии, минимизации, внешнего ветвления и, быть может, другие операции, вводимые пользователем.

Можно построить такую универсальную программу U, зависящую от $k + 1$ параметров, что, каково бы ни было заполнение базиса $Б_1$, для всех значений параметров $n, x_1, x_2, \ldots, x_k$ будет выполнено соотношение

$$u(n, x_1, x_2, \ldots, x_k) \simeq \varphi_n(x_1, x_2, \ldots, x_k).$$

Идея организации такой программы заключается, очевидно, в том, что если n — номер функции, принадлежащей базису $Б$, то значением U на соответствующем наборе $(n, x_1, \ldots, x_k)$ должно быть значение базисной функции с номером n на наборе $(x_1, \ldots, x_k)$, а если n — номер производной функции, то, анализируя этот номер, получаем дерево, висячие вершины которого соответствуют базисным функциям, а остальные вершины, в том числе и корень дерева, соответствуют производным функциям, каждая из которых порождается из более элементарных с помощью одной из трех основных операций (корень этого дерева соответствует функции α). Описанный процесс позволяет вычислять U на всех наборах, на которых значение U определено.

Организация работы системы строится следующим образом.

Этап 1. Функции (программы) $G_1, G_2, \ldots, G_p$ помещаются на первые свободные номера из $Б_1$, отличные от 1-го. Обозначим эти номера через $g_1, g_2, \ldots, g_p$.

Этап 2. В правой части описания функции α заменим все вхождения G_l на вхождения чисел g_l, а все вхождения α — на вхождения φ_l.

Этап 3. На основе описанного метода нумерации начинает работать алгоритм, который по любому числу q и по схеме R вычисляет

номер той функции, которая получится, если в полученной после Этапа 2 правой части описания заменить все вхождения φ_i на конкретную функцию с номером q. Обозначим функцию, реализуемую этим алгоритмом, через f. Функция fh помещается в $Б_1$ на место с номером 1. Отметим, что функция f может быть построена так, чтобы не зависеть от заполнений $Б_1$. На этом этап 3 завершается.

Этап 4. На местах вида $h(t)$ помещаются функции, вычисляемые следующим образом. Начать вычисление $U(t, t, x_2, \ldots, x_k)$. Если вычисление завершается и в результате получается число e, то вычислять $U(e, x_1, \ldots, x_k)$. Если и это вычисление завершается, то его результат объявить результатом работы всей программы.

Доопределение функции U на номерах вида $h(t)$ и производных от него не представляет никакой трудности.

В той части задачи, которая рассматривается в данной статье, этап 4 используется исключительно в варианте $t = 1$.

При выполнении этапа 4 может возникнуть как порочный круг в вычислениях, так и «разбегание» точек вычисления, в результате которых программа может не завершить свою работу. Но во всех случаях, представляющих для нас интерес, вычисления должны завершаться, поскольку мы предполагаем, что решение объективно существует и наша задача — найти его.

Убедимся теперь в том, что $h(1)$ есть неподвижная точка для функции f, то есть $\varphi_{fh(1)} = \varphi_{h(1)}$. В самом деле вычисление $\varphi_{h(1)}(x_1, x_2, \ldots, x_k)$ состоит в том, что сначала идет процесс вычисления значения $U(1, 1, x_2, \ldots, x_k)$. Но $U(1, 1, x_2, \ldots, x_k)$ есть значение функции, имеющей в нумерации номер 1, на наборе $(1, x_2, \ldots, x_k)$. На месте 1 в нумерации помещена функция fh, следовательно, $U(1, 1, x_2, \ldots, x_k) = fh(1)$, так как остальные переменные в функции fh фиктивны. Поскольку функция fh, очевидно, всюду определенная, то, следовательно, процесс вычисления $U(1, 1, x_2, \ldots, x_k)$ рано или поздно завершится и выдаст некоторый результат e. Следовательно, $fh(1) = e$. Итак, имеем

$$\varphi_{h(1)}(x_1, x_2, \ldots, x_k) = U(e, x_1, x_2, \ldots, x_k) = \varphi_{fh(1)}(x_1, x_2, \ldots, x_k).$$

Следовательно, $h(1)$ в действительности есть неподвижная точка для функции f, то есть

$$\varphi_{h(1)} = \varphi_{fh(1)} = P(\varphi_{h(1)}, G_1, G_2, \ldots, G_p).$$

Поскольку функция $\varphi_{h(1)}$ удовлетворяет условию, наложенному на α, то на этом синтез программы вычисления функции α завершается.

Мы не ставили задачи какой-либо оптимизации процесса синтеза и, естественно, вполне можно ожидать, что порождаемая программа может оказаться не из лучших. Заметим также, что условия, налагаемые на α, могут оказаться невыполнимыми, или решение для α может

оказаться не всюду определенной функцией. В этих случаях «вина» лежит на задании, а не на неудаче синтеза. Отметим также, что, к сожалению, по заданию α в принципе невозможно распознать окажется ли решение частично определенной функцией или нет. Более строго, нетрудно убедиться в том, что проблема распознавания свойства решения уравнения $\alpha = R(\alpha, G_1, G_2, ..., G_p)$ относительно α быть всюду определенной, частично определенной или нигде не определенной функцией, является алгоритмически неразрешимой проблемой.

Синтез программ методом, описанным выше, освобождает программиста от составления алгоритма вычисления, что экономит труд. Другой стороной преимуществ синтеза является то, что, доказав правильность функционирования системы, мы обеспечиваем автоматически правильность синтезированной программы, то есть соответствие поставленным условиям, если последние корректны. Отпадает необходимость отладки программы.

Ряд алгоритмов и программ, разработанных в русле описанной системы синтеза, представлен в работе [6], публикуемой в этом же сборнике.

Мы надеемся, что разработанная система окажется полезным инструментом для решения уравнений в программах.

Հ. Ռ. ՄԱՐԱՆՋՅԱՆ

ԹՎԱՅԻՆ ՖՈՒՆԿՑԻԱՆԵՐԻ ԾՐԱԳՐԵՐԻ ՍԻՆԹԵԶԻ ՄԻ ՄԵԹՈԴԻ ՄԱՍԻՆ

Հոդվածում առաջարկվում են Ս. Կ. Քլինիի ռեկուրսիայի երկրորդ թեորեմի հիման վրա մշակված ոչ բացահայտ ձևով տրված բնական փոփոխականի ֆունկցիաների հաշվման ընթացակարգի և ԷՀՄ-ի համար ծրագրի սինթեզի մի եղանակ։ Այդ եղանակի օգնությամբ հնարավոր է դառնում այնպիսի հավասարումների լուծումը, որոնց անհայտը՝ ծրագիր է։

ЛИТЕРАТУРА

1. *Барздинь Я. М.* Один подход к проблеме индуктивного вывода. Тезисы докладов III конференции «Применение методов математической логики», Таллин, 1983, 16—28.
2. *Ершов А. П.* Смешанные вычисления: потенциальные применения и проблемы исследования. Тезисы докладов и сообщений к Всесоюзной конференции «Методы математической логики в проблемах искусственного интеллекта и систематического программирования», 2, Вильнюс, 1980, 26—55.
3. *Клини С. К.* Введение в метаматематику, М., ИЛ, 1957.
4. *Минц Г. Е.* Логические основы синтеза программ. Препринт, Таллин, 1982.
5. *Непейвода Н. Н., Свириденко Д. И.* Программирование с логической точки зрения. Препринт, 1, 2, Новосибирск, Институт математики СО АН СССР, 1981.
6. *Петросян А. А.* Основные базисные функции для автоматизированного синтеза программ. Наст. сборник.

7. *Роджерс Х.* Теория рекурсивных функций и эффективная вычислимость, М., Мир, 1972.
8. *Тыугу Э. Х.* Синтез программ (обзор). Тезисы докладов и сообщений Всесоюзной конференции «Методы математической логики в проблемах искусственного интеллекта и систематическое программирование», 2, Вильнюс, 1980.
9. *Manna Z.* Mathematical Theory of Computation, New York, McGraw-Hill, 1974.
10. *Manna Z., Waldinger R.* Synthesis: Dreams ⟹ Programs. IEEE Transactions on Software Engineering, v. SE-5, 1979, № 4, 294—328.
11. *Martin-Löf P.* Constructive mathematics and Computer Programming. 6-th International Congress for Logic, Methodology and Philosophy of Science, 1979.